A developmental agent for learning features, environment models, and general robotics tasks

Brandon Rohrer
Intelligent Systems, Robotics, and Cybernetics Group
Sandia National Laboratories
Albuquerque, New Mexico, USA
Email: brrohre@sandia.gov
Web: http://www.sandia.gov/˜brrohre

***Abstract*—BECCA, a developmental agent, is described and demonstrated performing a high-dimensional visual servoing task. BECCA learns 1) a feature representation of its state space, 2) a model of its environment, and 3) how to behave in order to receive reward. It learns these things concurrently in an online and incremental fashion, without any prior knowledge of its environment or the nature of its inputs and outputs.**

I. Introduction

Biological developmental agents, such as children, learn both feature representations and world models through their actions and interactions. In this paper I describe a computational developmental agent that uses hypothesized biological mechanisms to learn a feature representation and a world model while performing a reward-based task.

Learning a *feature representation* is the act of mapping low-level inputs onto higher level perceptual symbols or categories. For instance, a set of pixels may be categorized as "an image of a puppy," which categorization may then be passed to modeling and action selection processes. On a more fundamental level, in the primary visual area of the mammalian neocortex, V1, small groups of inputs may be mapped onto line segments with strong orientation, spatial frequency, and directionality of motion. These features may then be passed to other areas of the brain that perform additional feature mapping and eventually action selection.

Learning a *world model* is the act of recording observed features in order to capture salient aspects of the agent's experience. Episodic, semantic, and procedural memory can all be considered aspects of a world model. Episodic memory contains specific instances of experience, while semantic memory condenses many experiences into a more general sense of meaning. Procedural memory is also a condensation of experience, but in the context of repeated motor actions.

The learning of feature representations and world models that are both useful and biologically plausible are among the chief technical challenges for those seeking to create developmental agents. [8] There are many research programs in this area that have successfully addressed aspects of these problems. ([6], [5], and [18] for example) This work is an effort to contribute to this body of research, addressing the problems of integrated feature, model, and task learning in a unified framework.

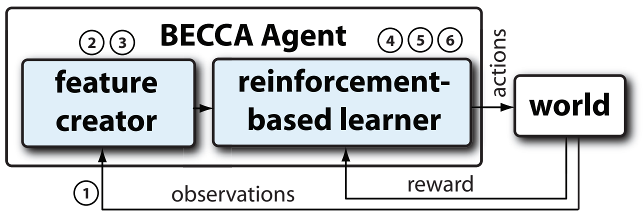

Fig. 1. At each timestep, the BECCA agent completes one iteration of the sensing-learning-planning-acting loop, consisting of six major steps: 1) Reading in observations and reward. 2) Updating its feature set. 3) Expressing observations in terms of features. 4) Predicting likely outcomes based on an internal model. 5) Selecting an action based on the expected reward of likely outcomes. 6) Updating its world model.

The agent design presented here is loosely based on the structure and function of the human brain and is referred to optimistically as a brain-emulating cognition and control architecture (BECCA). The remainder of the paper contains an algorithmic description of BECCA, an illustration of its operation on a simple task, and a discussion of its capabilities and limitations.

II. Method

A BECCA agent interacts with the world by taking actions, making observations, and receiving reward. (See Figure 1.) Formulated in this way, natural world interaction is a general reinforcement learning (RL) problem, [25] and BECCA is a potential solution. Specifically, at each discrete time step, BECCA performs three functions:

- It reads in an observation, a vector $\mathbf{o} \in \mathbb{R}^m | 0 \leq o_i \leq 1$.
- It receives a reward, a scalar $r \in \mathbb{R} | -1 \leq r \leq 1$.
- It outputs an action, a vector $\mathbf{a} \in \mathbb{R}^n | 0 \leq a_i \leq 1$.

Because BECCA is intended for use in a wide variety of environments and tasks, it can make very few assumptions about them beforehand. Although it is a model-based learner, it must learn an appropriate model through experience. BECCA uses two key algorithms to do this: an unsupervised feature creation algorithm (See Algorithm 1 and Figure 2.) and a tabular model construction algorithm (See Algorithm 2 and Figure 3).

The feature creation algorithm identifies patterns in the agent's input that are repeated and thus likely to have semantic

Algorithm 1 FEATURE CREATOR

```
Input: observation vector
Output: feature_activity vector

1:  form input vector by concatenating observation
        and previous feature_activity
2:  update estimate of correlation between inputs
3:  if MAX(correlation) $> C_1$ then
4:      add the two input elements achieving the
            maximum correlation to the new group
5:      while NOT(stop_condition_met) do
6:          find mean_correlation between each
                remaining input and group members
7:          if MAX(mean_correlation)$> C_2$ and
                group size $< C_3$ then
8:              add the input element to the group
9:          else stop_condition_met
10: for each group do
11:     if MIN (DISTANCE (input, features))$> C_4$ then
12:         add normalized input to set of features
13:     for each feature do
14:         feature_vote = feature · input
15:     feature_activity = WTA(feature_vote)
```

Algorithm 2 REINFORCEMENT LEARNER

```
Input: feature_activity vector
        reward scalar
Output: action vector

1:   attended_feature = SIGN(ARGMAX(feature_activity))
2:   decay working_memory and add attended_feature
3:   update model, consisting of
             cause–effect transition pairs
3.1:     cause = plan + previous_working_memory
3.2:     effect = attended_feature
3.3:     effect_matches = all transition pairs in model
             with matching effect
3.4:     if MIN(DISTANCE(effect_matches, cause)) $> C_5$
3.5:         add new cause–effect pair to the model
3.6:     else
3.7:         increment count of nearest cause–effect pair
4:   get predictions from model
4.1:     for each transition pair in model do
4.2:         weighted_effect = effect×
                 SIMILARITY(cause, working_memory)
5:   select action
5.1:     expected_reward =
             weighted_effect × reward_map
5.2:     find cause–effect pair associated with
             MAX(expected_reward)
5.3:     plan = cause from the cause–effect pair
5.4:     action = motor portion of the plan
5.5:     on a fraction, $C_6$, of time steps,
             generate a random exploratory action
6:   update reward_map using feature_activity
         and reward
```

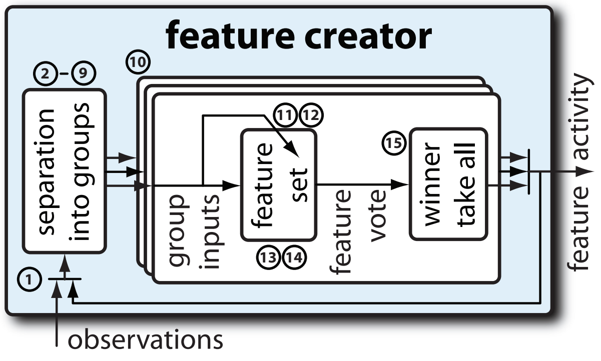

Fig. 2. Block diagram of the feature creator, illustrating its operation. Numbered labels refer to steps in Algorithm 1.

relevance. It works by grouping the elements of the input vector into groups whose activity is somewhat correlated. In a pixel array exposed to a video stream of broadcast television, for example, the correlation between two neighboring pixels will be much higher than that occurring between distant pixels, and a small number of pixels grouped by correlation will be closely related in space. The groups of input elements form input subspaces, and unit vectors in these subspaces represent features . The feature creator creates new features by adopting novel inputs, also known as imprinting. Inputs must be sufficiently different from existing features in order to be imprinted. Returning to the pixel array example, once a small group of correlated pixels has been formed, features will be created based on patterns observed in those pixels. These may include horizontal, vertical, and diagonal edges, (as in Figure 6 as well as uniform intensity and center-surround patterns.

In addition to creating and updating the feature set at each time step, the feature creator projects the inputs onto exiting features to calculate feature votes. These feature votes are then subjected to a winner-take-all operation, such as might be implemented in a neural network with mutual inhibition. A single feature in each group remains active, and the set of active features is passed on to the reinforcement learner. It is also fed back and combined with the next observation to form the input for the next time step. The recursive nature of the feature creation algorithm allows more complex features to be created from combinations of simpler ones.

The reinforcement learner takes in feature activity and reward and selects an action to execute. The reward map associates features with reward by approximating the correlation between reward and each feature. An attention filter selects the most salient feature at each time step as the attended feature. Working memory is a weighted combination of several recent attended features and any recent actions. The attended feature and working memory are used to update the model. The model is a table of cause–effect pairs, where each effect is an attended feature and each cause is the working memory from the preceding time step. The model also contains a record of the number of times each pair is observed. Rarely observed pairs are periodically removed. This table of cause–effect pairs provides a record of common transitions in feature space, as well as any actions that may have been taken to precipitate them.

In order to make predictions, the current working memory is matched against causes in the model. The corresponding set

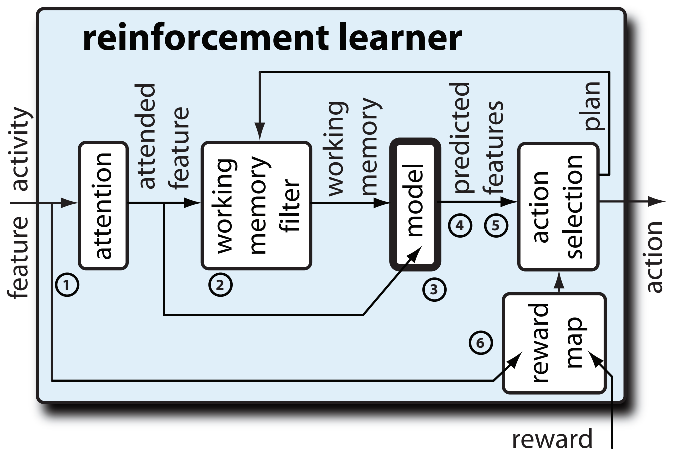

Fig. 3. Block diagram of the reinforcement learner, illustrating its operation. Numbered labels refer to steps in Algorithm 2.

of likely effects are predictions. Predicted effects with high expected reward are found, and the action associated with the highest reward is selected. Occasionally a random exploratory action is substituted for the greedy action.

III. Results

A simulated task similar to the Morris water maze [17] was constructed to demonstrate BECCA in operation. (See Figure 4.) An agent panned a virtual camera vertically and horizontally about an image. The camera produced a 10 $\times$ 10 pixel virtual image of its field of view, with each pixel taking on a value between 0 (black) and 1 (white). Each observation comprised the 100 pixel values, v, as well as their complement, $1-v$. The action vector represented four discrete movement steps in each direction with magnitudes of $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, and $\frac{1}{16}$ of the width of the field of view. When concatenated, each observation and action resulted in a 216 element vector, which was passed to BECCA at each time step. BECCA was rewarded for directing its gaze at the center of the mural. It learned to do this reasonably well after approximately 12,000 time steps. Complete MATLAB code for the simulation and BECCA implementation (version 0.3.5) can be found at [20]. The simulation required approximately 55 minutes to complete on a 64-bit 2.9GHz Intel Core i7 processor running MATLAB 2010a under Windows 7, however, very little has been done to optimize the code for speed. When saved after completion, the agent's backup file occupied 3.23 MB.

The majority of the 12,000 time steps was spent learning a feature representation of the inputs. Most of this time was required for the correlation estimate to reach a sufficiently high level to nucleate groups. (See Figure 5.) The correlation estimate was calculated on line and incrementally. In order to reduce noise and avoid spurious groupings, it was incremented very gradually. After a group was created, features were learned within several time steps, (See Figure 6.) and the system model was learned after that, reaching a size of 574 cause–effect pairs. (See Figure 7.) In fact, all aspects of learning (groups, features, and model) were ongoing throughout the simulation, but individual aspects were most prominent at different times, demarcating stages of learning.

IV. Discussion

In the visual servoing task, BECCA demonstrated its ability to achieve better than random performance on a RL task with a 216-dimensional observation-action space, about which it had no prior knowledge.

The visual servoing task is trivial in the sense that it has many straightforward solutions that incorporate some knowledge of the task. For instance, after studying the task, a human designer could hand-select useful features and rules for how to behave when those features were observed. The agent custom designed in this way would likely learn faster and perform better than BECCA in a direct comparison, but such a comparison would be misleading. In the case of the task-customized agent, a human performs the feature creation and policy construction functions that BECCA handled itself. Thus BECCA would not be compared to the task-customized agent, but rather to the agent-designer team.

BECCA was designed to handle a set of natural interaction tasks that is as broad as possible. The visual servoing task did not adequately illustrate this breadth; it was selected as a means to illustrate BECCA's operation as simply as possible. It should be noted that, simple as it is, a problem with a 216-dimensional state space is considered very challenging or infeasible for many learning methods.

A. Design decisions: Reinforcement learning and embodiment

Reward-based tasks were selected as the focus for this work because of their similarity to the tasks addressed by biological developmental agents. Some developmental approaches focus on narrower tasks, such as determining class membership of inputs (as in supervised learning [21]) or world modeling (as in artificial curiosity, e.g. [13], [11]). In behaving biological organisms, these tasks are a means to an end. Machine leaning methods focused on narrow aspects of learning, such as perception, classification, clustering, and learning production rules, all require a supporting framework in order to be applied to produce goal-directed behavior. The ability to categorize inputs or to model the world can help achieve these goals, but do not translate directly to success in them. Conversely, in some tasks it is possible to be highly successful in achieving reward-based goals without learning to distinguish many classes of inputs or completely model the world. Reward can be assigned to any essential activity, such as finding food, avoiding predation, and mating. The goals of categorization and modeling may be neither necessary nor sufficient for achieving these goals. In the general RL problem formulation used here, only the dimensionality of the observation and action spaces are known beforehand, and the agent seeks to maximize its reward. This framework is roughly descriptive of human and animal agents acting in their environments, suiting it to describe the problem of natural world interaction for

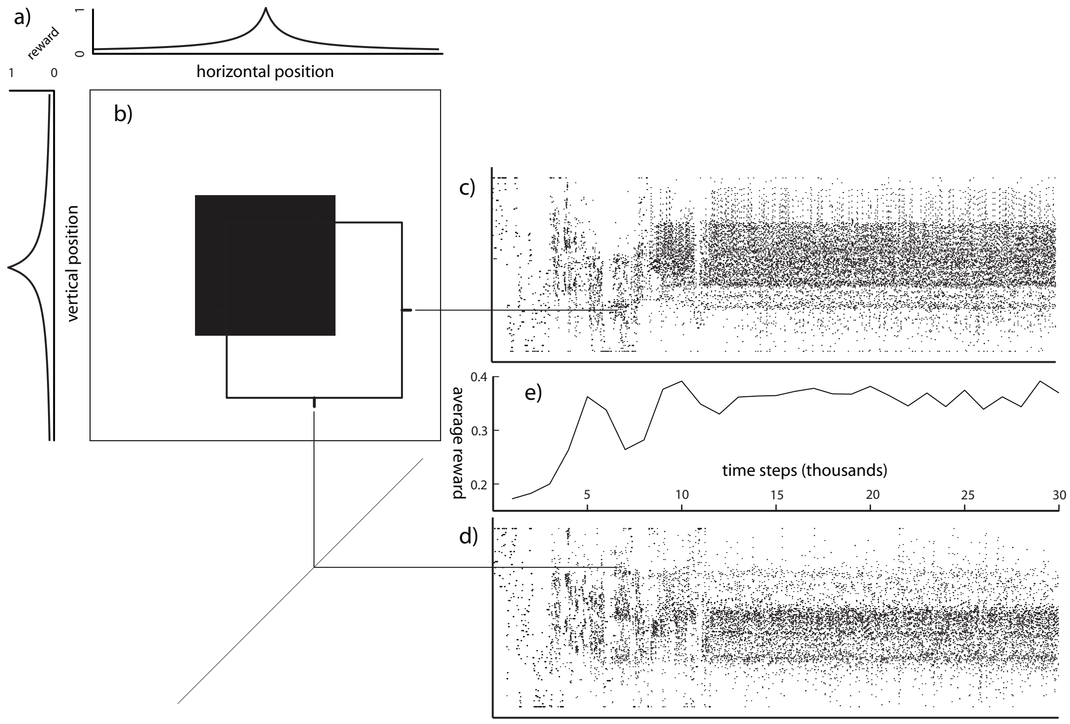

Fig. 4. A Morris water maze-type two-dimensional visual servoing task. **a)** The reward signal as a function of the agent's gaze position. It is 1 at the center of the image and falls off as $1/d$, where d is the distance of the gaze position from the center of the image. **b)** The task environment. The white field with black square provided a visual world. The frame representing the agent's field of view and the location of its gaze is also shown. The agent executed horizontal and vertical panning movements to adjust its gaze position. **c)** The agent's gaze position history, vertical component. After approximately 12,000 time steps the gaze focused more on the high reward region. **d)** The agent's gaze position history, horizontal component. **e)** Average reward per time step. After approximately 12,000 time steps the average reward stabilized to roughly twice its initial value.

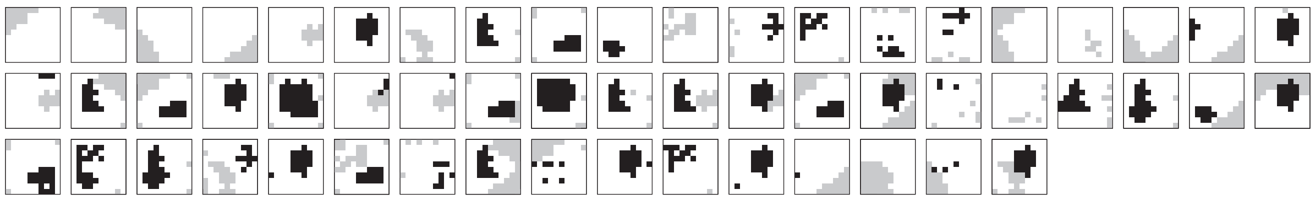

Fig. 5. Receptive fields for each of the 56 groups. The inputs selected for inclusion in each group are shown here. Light-responsive inputs are gray, and dark-responsive inputs are black. The tight spatial groupings of pixels resulted from their correlation and not from any information about their location in the array. For the most part, the groups in the top row are of individual pixels. The groups in the lower two rows are mostly higher level combinations of features from top row groups.

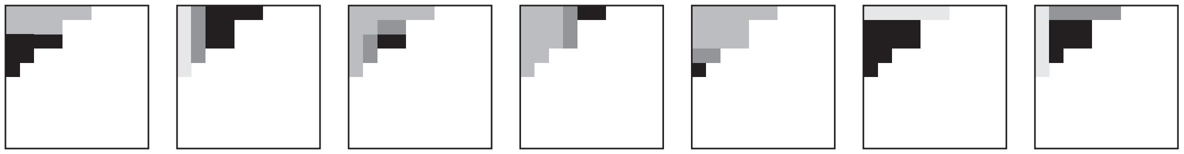

Fig. 6. Features created from the first group in Figure 5. These features primarily show horizontal and vertical edges at varying positions. The third feature also shows a corner. The nature of these features follow intuitively from the position of the group; it is expected that the upper-left hand corner of the field of view should be exposed to the top edge, the left edge, and the upper-left hand corner of the black box in the image.

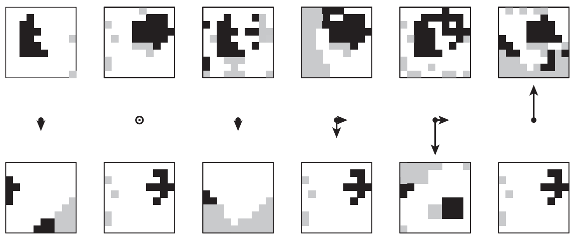

Fig. 7. The six most common sequences recorded within the model, shown with cause above and effect below. Each arrow indicates the direction of gaze shift and results in image motion in the opposite direction. Because the cause–effect pairs are represented in terms of the agent's internal features, they can be difficult to intuit.

machine agents as well. Within such a general RL framework, more specialized algorithms may then be used in concert to most effectively seek out reward. For example, artificial curiosity has been very effectively incorporated into a reward-driven task. [22]

Although the visual servoing task was not implemented in physical hardware, the design of BECCA has been conducted to suit it for use on physical robots in unstructured environments. The emphasis on embodiment [19] and unstructured task environments [28] is a recurring theme in developmental robotics. Despite the daunting scope of the problem, recent work in this direction has produced inspiring results. [23], [26] Current work is focused on integrating BECCA with physical robots to perform tasks in unstructured environments.

While BECCA has been designed using insights gleaned from experimental psychology and cognitive neuroscience, it is not intended to be a cognitive or neural model of how a human or animal brain operates. BECCA incorporates computational mechanisms, such as the dot product or winner-take-all, that have plausible neuronal implementations, but there is no associated claim, express or implied, that the brain actually performs those calculations. Put simply, BECCA's purpose is not to describe the brain, but to perform like it.

B. Related work: Unsupervised learning

BECCA's feature creation algorithm is an example of an unsupervised learning method, in that it learns a structure based only on the observed data. There are many other examples of algorithms that do this automatically, although none with the same properties as BECCA. Unsupervised feature creation has been shown to be a useful method for concept generation in developmental robots. [7], [4] There are many unsupervised learning methods developed with different sets of assumptions, [9] but BECCA's feature creator provides a novel collection of characteristics. It is *on-line*, meaning that it incorporates data points one at a time and modifies its feature representation incrementally. It is *hierarchical* in that it can use created features to construct still higher level features. It is *stable* in the sense that feature definitions do not change once they are created. In contrast to many unsupervised learning algorithms, BECCA's feature creator does not assume the number of features that exist in the underlying data. Once the dimensionality of the state space is defined, the feature creator always starts from the same initial conditions, so there is no need to carefully pick initial values for cluster parameters. Only four thresholding constants need to be selected. Like other unsupervised learning methods in which the number of clusters is not specified, the validity of the features found depends entirely on the appropriateness of the measure of feature goodness. And, as with other unsupervised learning algorithms of its class, there are no theoretical performance guarantees.

C. Related work: Deep learning

The problem of hierachical feature creation is closely related to deep learning. [3] Deep learning approaches seek to discover and exploit the underlying structure of a world by creating higher level, lower-dimensional representations of the system's input space. Deep learning algorithms include Convolutional Neural Networks (CNN) [14], Deep Belief Networks (DBN) [12], [10], and the Deep SpatioTemporal Inference Network (DeSTIN) [2]. Deep learning algorithms such as these are alternative approaches, worthy of consideration for automatic concept acquisition, although they differ somewhat from BECCA's feature creator. CNNs are designed to work with two-dimensional data, such as images, and they do not apply to arbitrarily structured data, as BECCA does. By using several layers of Restricted Boltzmann Machines, DBNs are capable of generating sophisticated features that allow it to interpret novel inputs. However, they are typically applied to the supervised learning problem of discrimination, and require a substantial amount of labeled data in order to be adequately trained. Whether DBNs can be applied to the unsupervised learning problem of feature creation is unclear. DeSTIN incorporates both unsupervised and supervised learning methods and appears to be fully capable of hierarchical feature creation. It has been published only recently; future papers describing its operation and performance will allow a more detailed comparison with BECCA's feature creator.

D. Related work: Reinforcement learning

There are several well known methods for solving the reinforcement learning problem, that is, given state inputs and a reward at each timestep, choose actions that maximize the future reward. Some examples that have been applied in agents include Q-learning [27], the Dyna architecture [24], Associative Memory [15], and neural-network-based techniques including Brain-Based Devices [16] and CMAC [1]. BECCA's reinforcement learner is another such algorithm. It is *on-line* and *model-based*, meaning that as it accumulates experience it creates and refines an internal model of itself and its environment. It differs from most previous methods in two ways. First, its internal model is not a first order Markov model. Instead, by using cause-effect transition pairs in which the cause is a compressed version of the agent's recent state history, it creates a compressed higher order Markov model. This potentially allows BECCA to learn more sophisticated state dynamics and to record distinct sequences more naturally. Second, BECCA's reinforcement learning algorithm can handle a growing state space. This is necessary because it must work in tandem with BECCA's feature creator, which continues to identify new features throughout the life of the agent.

V. Conclusion

The problem of natural world interaction continues to challenge cognitive scientists, artificial intelligence practicioners, and developmental psychologists. How do we find food, shelter, and social acceptance in a complex world? How does a child learn how to run, speak, and play the Wii? Can we make machines capable of doing everything we can? This problem has resisted immediate solution. Progress to date falls into two categories: agents that perform narrow skills (such as playing chess or Jeopardy) extremely well and agents capable of learning broad classes of tasks. BECCA falls into the second category.

This work is a demonstration of several key developmental capabilities: feature learning, model learning, and reward structure learning. BECCA is particularly notable in that it performs these simultaneously, incrementally, on-line, and with no *a priori* knowledge. But this work is only a small step toward the grand goal of natural world interaction. The task was simple, the features created were crude, the agent's performance was modest, and everything took place in simulation. Current work is focused on mitigating these criticisms by addressing more challenging tasks, integrating more aspects of human information processing, and embodying the agent in physical robot hardware. If successful, this will result in an ever more capable developmental agent.

Acknowledgements

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] J. Albus. A new approach to manipulator control: Cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement and Control*, 97:220–227, 1975.

[2] I. Arel, D. Rose, and B. Coop. DeSTIN: A deep learning architecture with application to high-dimensional robust pattern recognition. In *Proc. AAAI Workshop Biologically Inspired Cognitive Architectures (BICA)*, 2009.

[3] I. Arel, D. C. Rose, and T. P. Karnowski. Deep machine learning—a new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, November 2010.

[4] I. Atil, N. Dag, S. Kalkan, and E. Sahin. Affordances and emergence of concepts. In *Proceedings of the Tenth International Conference on Epigenetic Robotics*, 2010.

[5] C. Balkenius, J. Morn, B. Johansson, and M. Johnsson. Ikaros: Building cognitive models for robots. *Advanced Engineering Informatics*, 24(1):40–48, 2010.

[6] C. Balkenius and S. Winberg. *Tenth Scandinavian Conference on Artificial Intelligence (SCAI 2008)*, volume 173 of *Frontiers in Artificial Intelligence and Applications*, chapter Fast Learning in an Actor-Critic Architecture with Reward and Punishment, pages 20–27. IOS Press, 2008.

[7] P. Beeson, J. Modayil, and B. Kuipers. Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. *The International Journal of Robotics Research*, 29(4):428–459, 2010.

[8] A. Cangelosi, G. Metta, G. Sagerer, S. Nolfi, C. L. Nehaniv, K. Fischer, J. Tani, B. Belpaeme, G. Sandini, L. Fadiga, B. Wrede, K. Rohlfing, E. Tuci, K. Dautenhahn, J. Saunders, and A. Zeschel. Integration of action and language knowledge: A roadmap for developmental robotics. *IEEE Transactions on Autonomous Mental Development*, 2(3):167–195, 2010.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, second edition, 2001.

[10] I. Fasel and J. Berry. Deep belief networks for real-time extraction of tongue contours from ultrasound during speech. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2010.

[11] I. Fasel, A. Wilt, N. Mafi, and C. Morrison. Intrinsically motivated information foraging. In *Proceedings of the 2010 International Conference on Development and Learning*, 2010.

[12] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:15271554, 2006.

[13] P.-Y. Oudeyer F. Kaplan and V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, 1995.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[15] S. E. Levinson. *Mathematical Models for Speech Technology*. John Wiley and Sons, Chichester, England, 2005. pp. 238-239.

[16] J. L. McKinstry, G. M. Edelman, and J. L. Krichmar. A cerebellar model for predicitive motor control tested in a brain-based device. *Proceedings of the National Academy of Sciences*, 103(9):3387–3392, 2006.

[17] R. G. M. Morris. Spatial localization does not require the presence of local cues. *Learning and Motivation*, 12(2):239–260, 1981.

[18] A. F. Morse, J. de Greeff, T. Belpeame, and A. Cangelosi. Epigenetic Robotics Architecture (ERA). *IEEE Transactions on Autonomous Mental Development*, 2(4):325–339, 2010.

[19] G. Pezzulo, L. W. Barsalou, A. Cangelosi, M. H. Fischer, K. McRae, and M. J. Spivey. The mechanics of embodiment: A dialog on embodiment and computational modeling. *Frontiers in Psychology*, 2, 2011.

[20] B. Rohrer. BECCA code page. http://www.sandia.gov/˜brrohre/code.html, 2011.

[21] J. Sinapov and A.Stoytchev. Object category recognition by a humanoid robot using behavior-grounded relational learning. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[22] A. Stoytchev and R. Arkin. Incorporating motivation in a hybrid robot architecture. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 8(3):269–274, 2004.

[23] V. Sukhoy and A. Stoytchev. Learning to detect the functional components of doorbell buttons using active exploration and multimodal correlation. In *Proceedings of the 10th IEEE International Conference on Humanoid Robots*, pages 572–579, Nashville, Tennessee, 2010.

[24] R. S. Sutton. Planning by incremental dynamic programming. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 353–357. Morgan Kaufmann, 1991.

[25] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, 1998.

[26] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, Taipei, Taiwan, 2011.

[27] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3-4):279–292, May 1992.

[28] J. Weng. Muddy tasks and the necessity of autonomous mental development. In *Proc. 2005 AAAI Spring Symposium Series, Developmental Robotics Symposium*, Stanford University, Mar 21-23 2005.